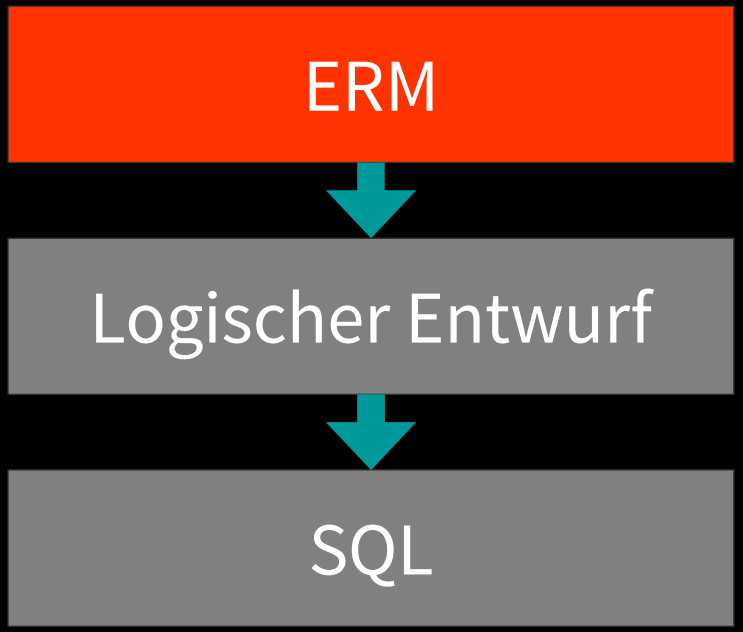

h da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
MIT3
DATENBANKEN
~2~
Google | google.com/datacenters
Claudius Coenen

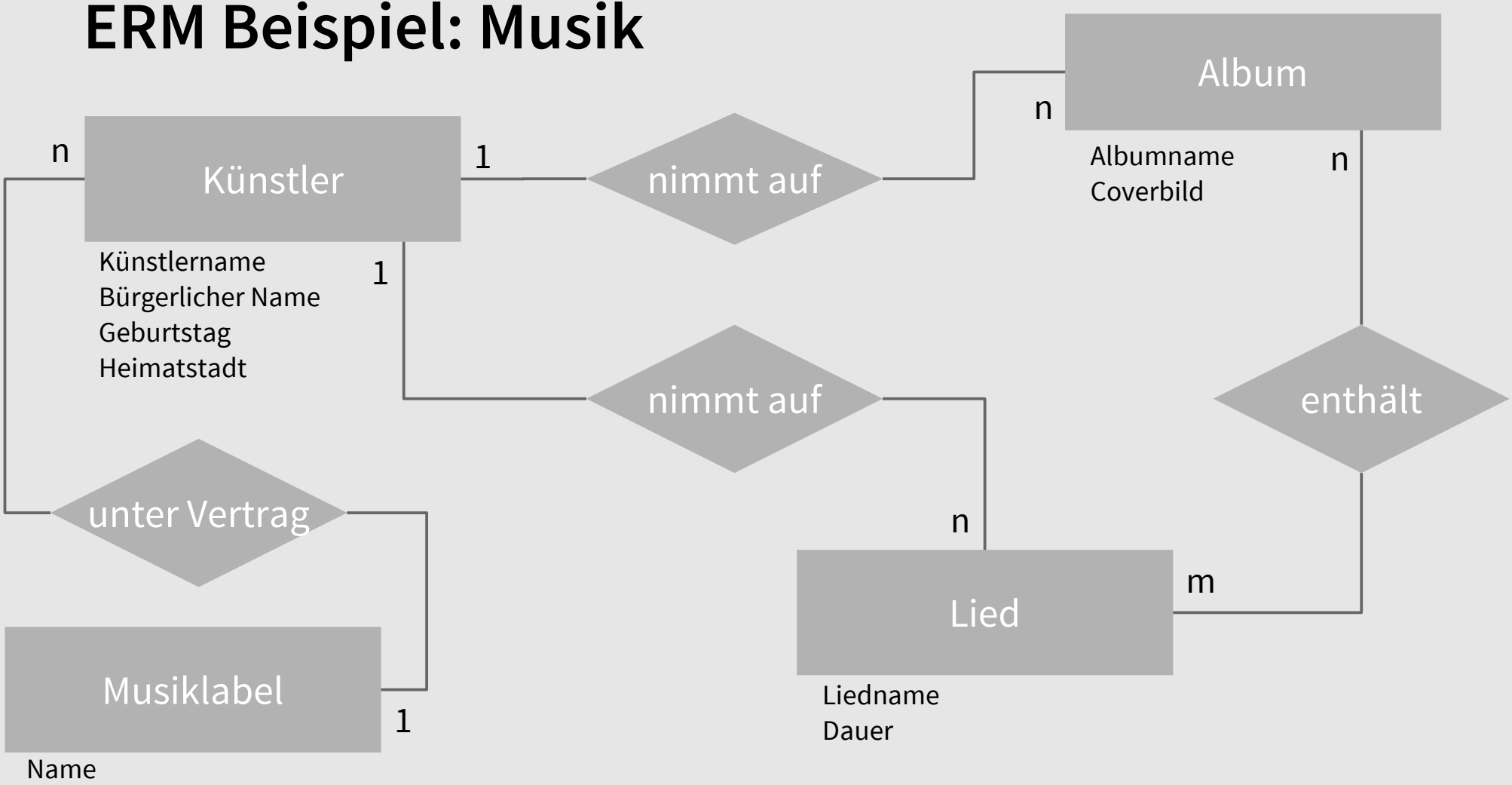


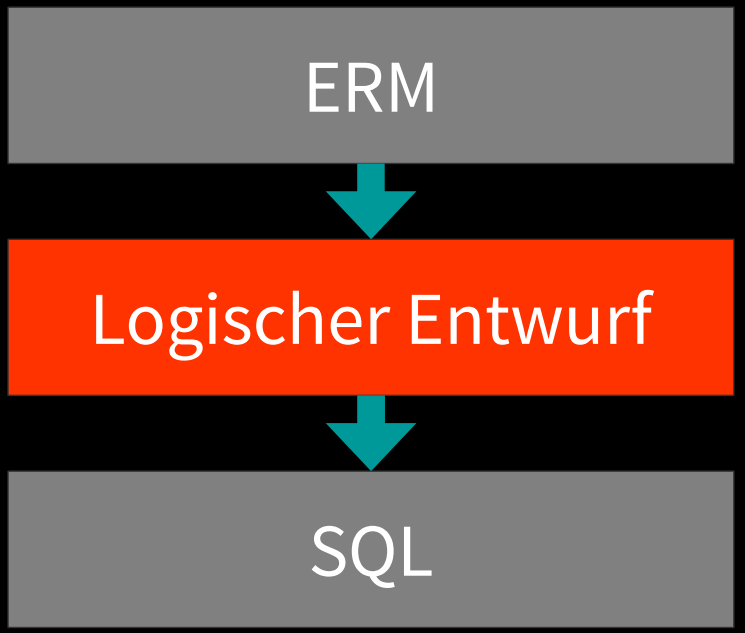

# (KUUURZE ERINNERUNG)

# Konzeptioneller Entwurf: Entity Relationship Model (ERM)

- Erstmal **Entitäten**
- Dann deren **Attribute**
- Die haben dann noch **Beziehungen**
- Konzeptionellen Entwurf ist schon fast Datenbank-Struktur

# ERM Beispiel: Musik




ERM
Logischer Entwurf
SQL

# ERM → Datenbankschema

## Attribute aus ERM

- musiclabel
  - name
  - established
- artist
  - artist_name
  - common_name
  - birthday
  - hometown
- album
  - name
  - cover
- track
  - name
  - duration

# ERM → Datenbankschema
## Datentypen

- musiclabel
  - name VARCHAR(255)
  - established DATE
- artist
  - artist_name VARCHAR(255)
  - common_name VARCHAR(255)
  - birthday DATE
  - hometown VARCHAR(255)

Auf den Attribut-Namen folgt der Datentyp.
Eine Liste der Typen kommt später!

- album
  - name VARCHAR(255)
  - cover BLOB
- track
  - name VARCHAR(255)
  - duration INT

# ERM → Datenbankschema
## IDs und Fremdschlüssel

- musiclabel
  - id INT AUTO_INCREMENT
  - name VARCHAR(255)
  - established DATE
- artist
  - id INT AUTO_INCREMENT
  - label_id INT
  - artist_name VARCHAR(255)
  - common_name VARCHAR(255)
  - birthday DATE
  - hometown VARCHAR(255)

IDs sind praktisch zum referenzieren.

„Ein Künstler ist bei einem Plattenlabel unter Vertrag“
Wir schreiben die ID des Plattenlabels in den Datensatz des Künstlers.

„Track / Album gehört genau zu einem Künstler“

- album
  - id INT AUTO_INCREMENT
  - artist_id INT
  - name VARCHAR(255)
  - cover BLOB
- track
  - id INT AUTO_INCREMENT
  - artist_id INT
  - name VARCHAR(255)
  - duration INT
- album_has_track
  - album_id INT
  - track_id INT

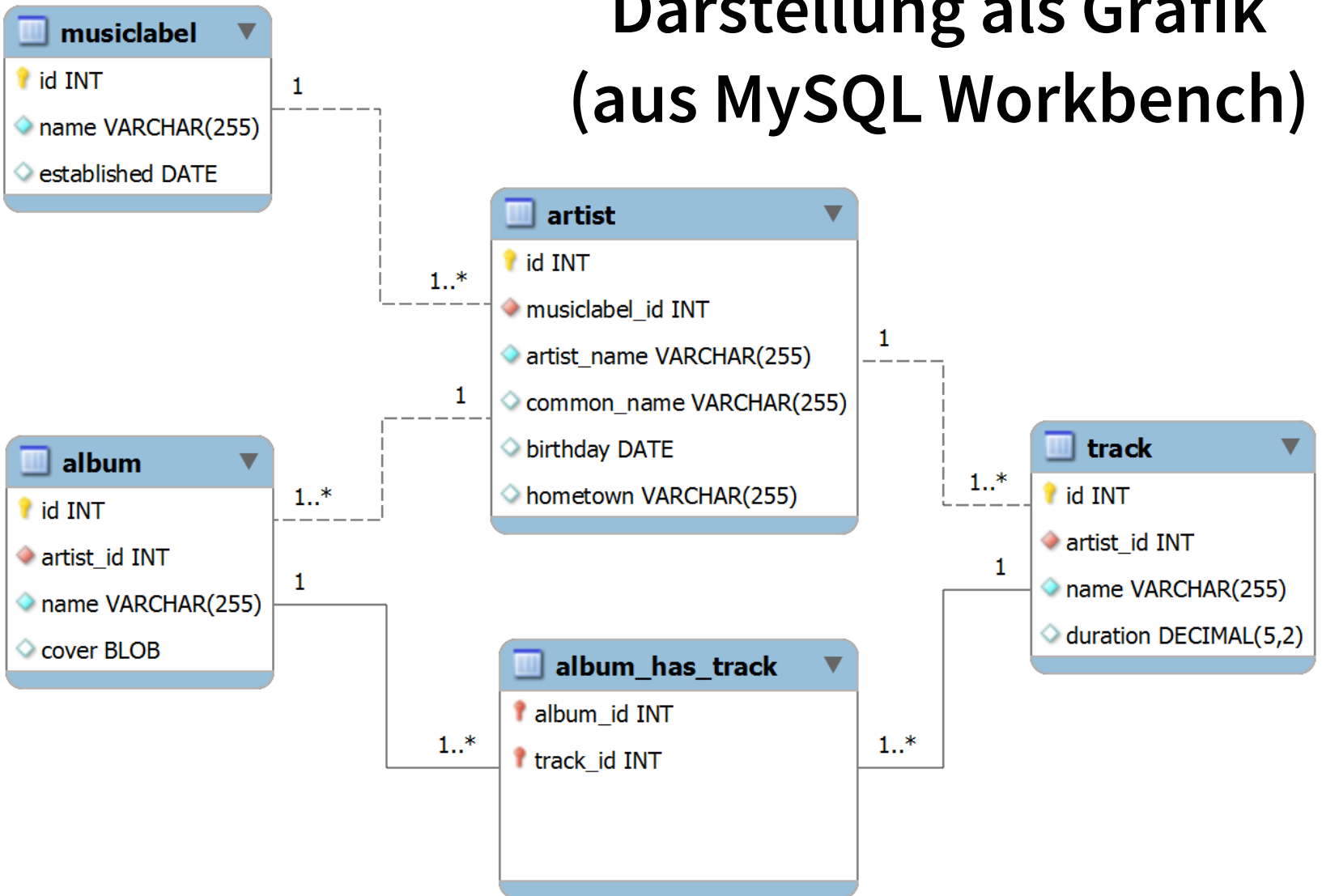
# ERM → Datenbankschema
## IDs und Fremdschlüssel

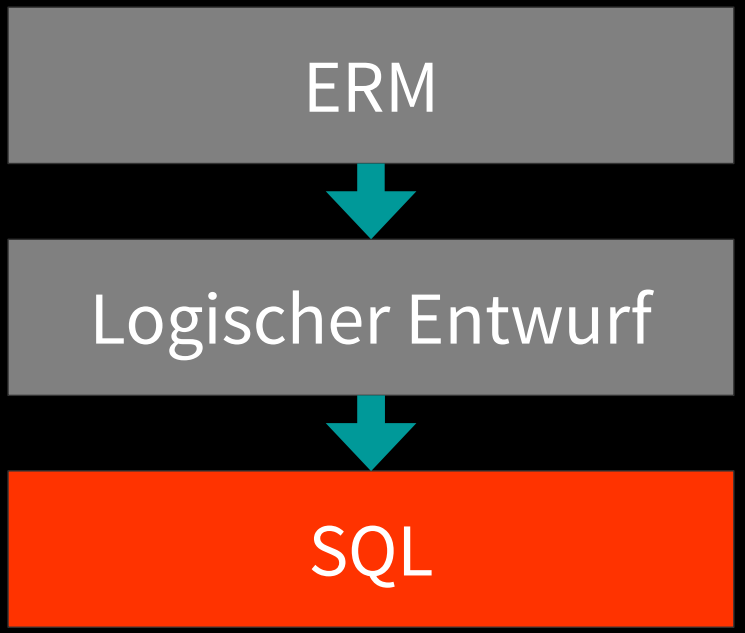
- musiclabel
  - id INT AUTO_INCREMENT
  - name VARCHAR(255)
  - established DATE
- artist
  - id INT AUTO_INCREMENT
  - label_id INT
  - artist_name VARCHAR(255)
  - common_name VARCHAR(255)
  - birthday DATE
  - hometown VARCHAR(255)

album_has_track ist ein sogenannter „Join-Table“. Er sorgt für die n:m Beziehung zwischen Album und Track.

- album
  - id INT AUTO_INCREMENT
  - artist_id INT
  - name VARCHAR(255)
  - cover BLOB
- track
  - id INT AUTO_INCREMENT
  - artist_id INT
  - name VARCHAR(255)
  - duration INT
- album_has_track
  - album_id INT
  - track_id INT

# Darstellung als Grafik (aus MySQL Workbench)

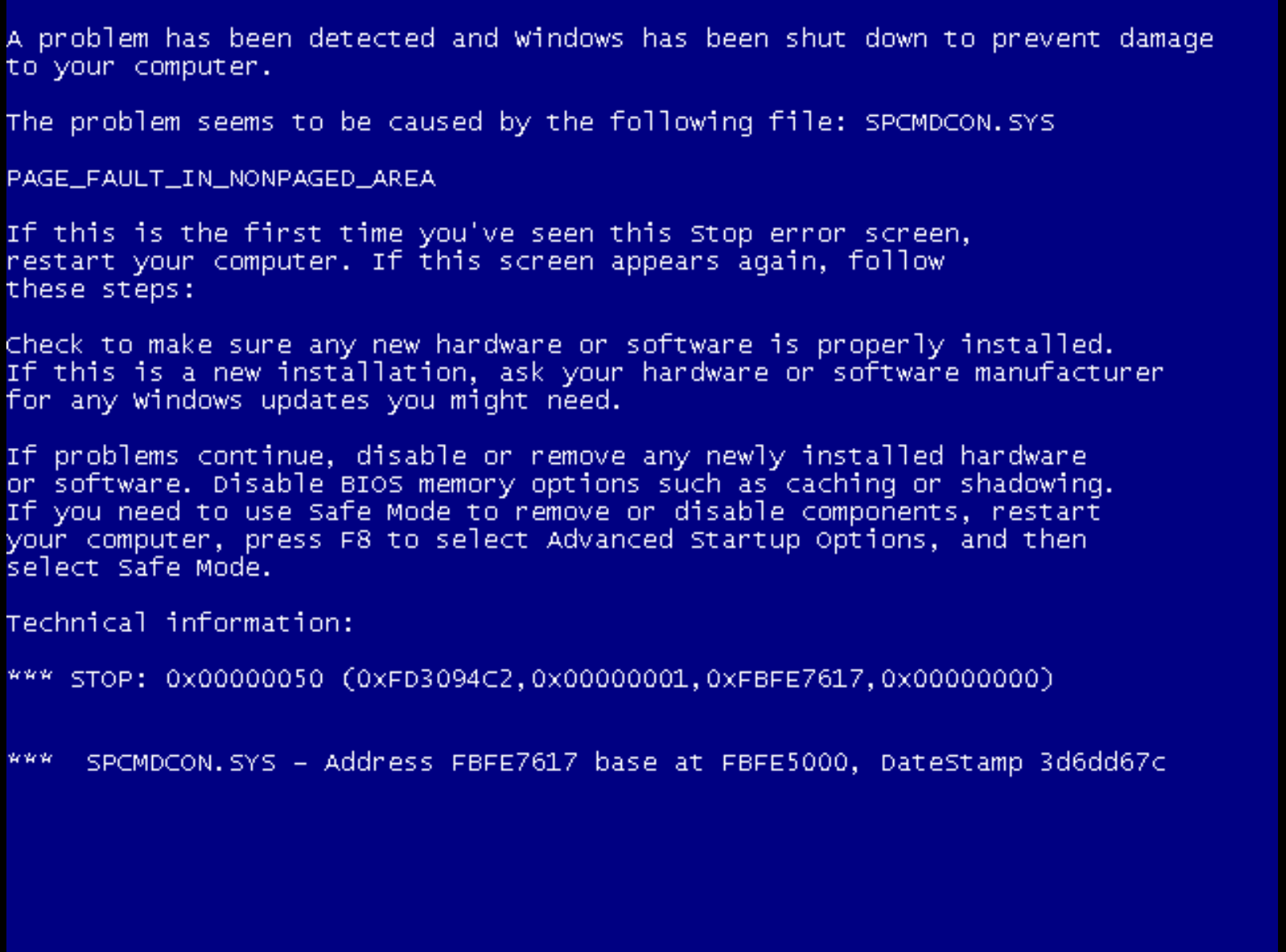



ERM
Logischer Entwurf
SQL

# Musik-Beispiel als SQL Befehle (nur als Beispiel!)

```sql
CREATE SCHEMA `music` ;
USE `music` ;

CREATE TABLE `musiclabel` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `established` DATE NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `artist` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `musiclabel_id` INT NOT NULL,
  `artist_name` VARCHAR(255) NOT NULL,
  `common_name` VARCHAR(255) NULL,
  `birthday` DATE NULL,
  `hometown` VARCHAR(255) NULL,
  PRIMARY KEY (`id`));
```

```sql
CREATE TABLE `album` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `artist_id` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  `cover` BLOB NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `track` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `artist_id` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  `duration` DECIMAL(5,2) NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `album_has_track` (
  `album_id` INT NOT NULL,
  `track_id` INT NOT NULL,
  PRIMARY KEY (`album_id`, `track_id`));
```


A problem has been detected and Windows has been shut down to prevent damage
to your computer.
The problem seems to be caused by the following file: SPCMDCON.SYS
PAGE_FAULT_IN_NONPAGED_AREA
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.
Technical information:
*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)
***  SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c

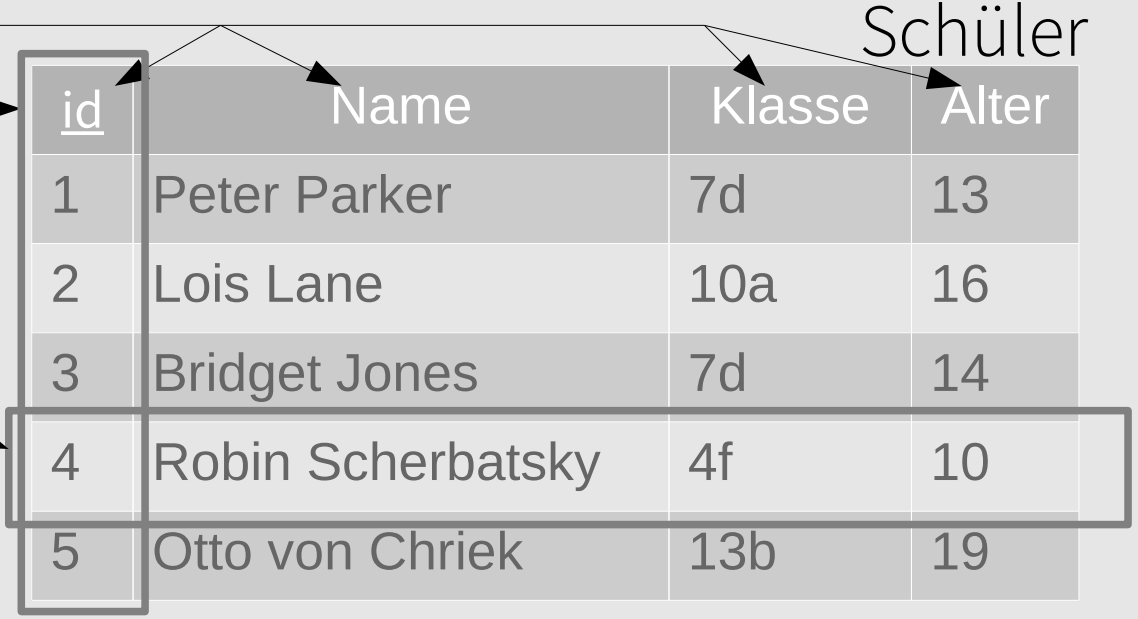
# SQL
# KURZREFERENZ

# SQL Begriffe

- Mit vielen DBMS spricht man „SQL“ - Structured Query Language
- SQL ist ein ISO Standard, viele DBMS erweitern die Sprache um eigene Funktionen
- Begriffe
  - Schema (eine Datenbank innerhalb von z.B. MySQL)
  - Relation bzw. Tabelle
  - Schlüssel / Primärschlüssel
  - Index
  - (einige davon nachfolgend erklärt)
- Guter Einstieg: http://de.wikipedia.org/wiki/SQL

# Tabelle

- Relation → umgangssprachlich „Tabelle“
- Tabellenname
- Attribut-Name
- Primärschlüssel
  → oft **id** oder ähnlich genannt
- Tupel oder Record
  → eine Zeile der Tabelle
- Attribute haben
  einen festen Datentyp



# Datentyp

- In SQL muss man sich im Vorfeld für Datentypen entscheiden
- Bestimmte Typen haben spezielle Hilfsfunktionen (zum Beispiel DATE)
- Grob aufgeteilt in
  - Numerische Werte
    → boolean, integer, numeric, decimal, float … jeweils signed oder unsigned
  - Zeit
    → date, time, datetime, timestamp …
  - Zeichenketten, Texte, Binärdaten
    → char, varchar, text, blob, enum
- Alle Datentypen können auch NULL enthalten (nicht „0“ als Ziffer, sondern NULL als Keyword)
- Ausführlich für MySQL: http://dev.mysql.com/doc/refman/5.6/en/data-types.html

# Schlüssel

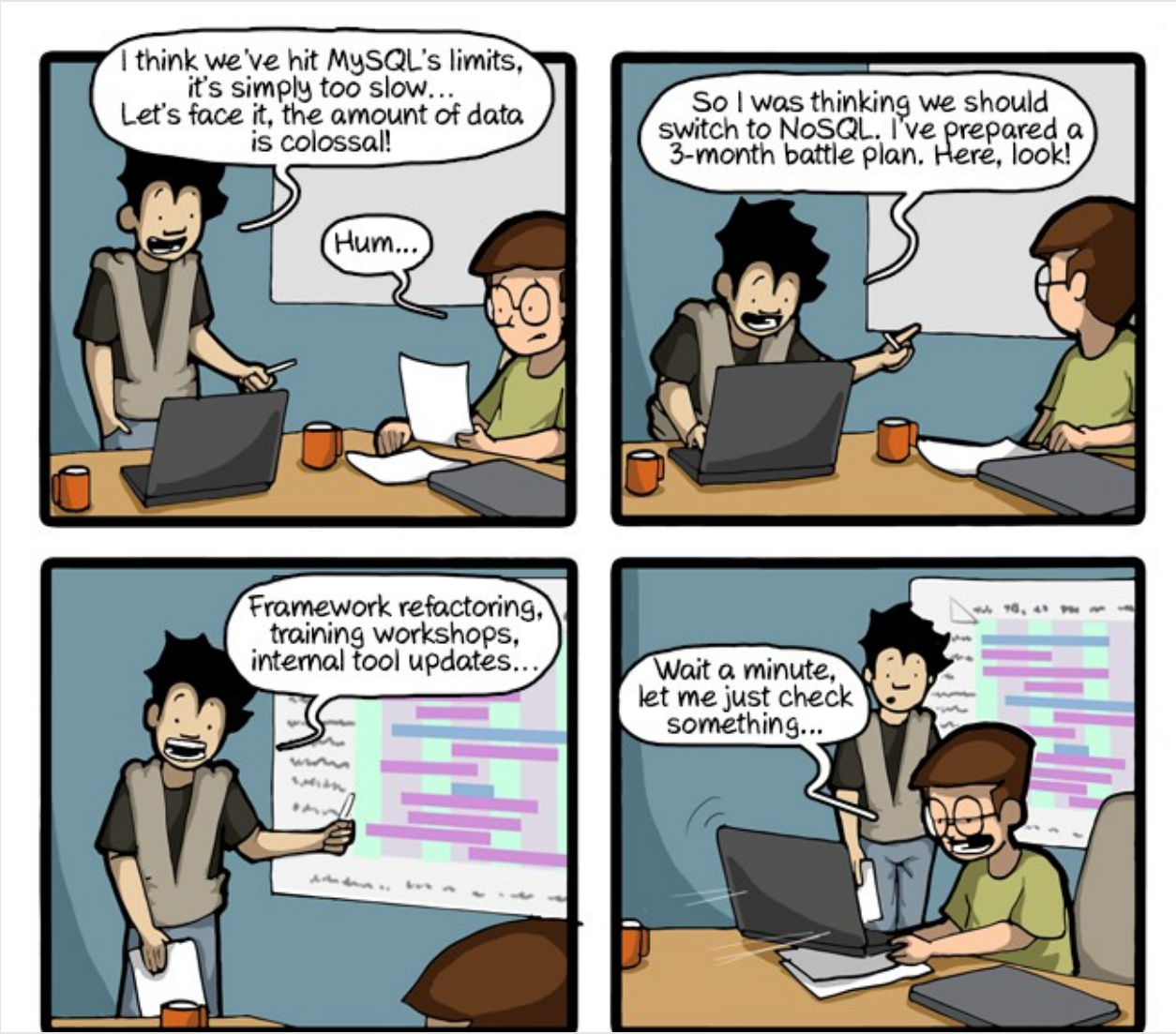
- Attribut(e) zur eindeutigen Identifikation eines Datensatzes
  → Kann auch über mehrere Attribute hinweg sein
- In der Praxis oft einfach eine ID nur zu diesem Zweck
- „Primärschlüssel“ ist derjenige, den wir als Identifikationsmerkmal aussuchen

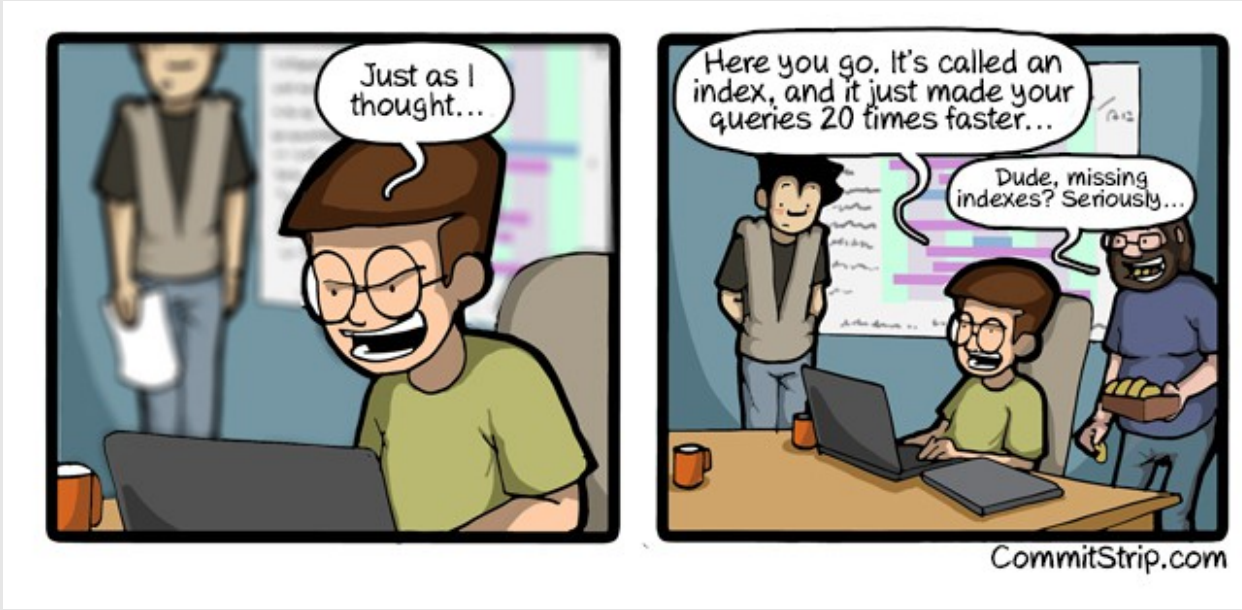

Schüler

| id | Name | Geburtstag | Handynummer | Klasse | Alter |
|---|---|---|---|---|---|
| 1 | Peter Parker | 2001-05-01 | 0172 123458 | 7d | 13 |
| 2 | Lois Lane | 1998-07-12 | 0151 345889 | 10a | 16 |
| 3 | Bridget Jones | 2000-07-13 | 0162 111222 | 7d | 14 |
| 4 | Robin Scherbatsky | 2004-09-21 | 0180 5 12 | 4f | 10 |
| 5 | Otto von Chriek | 1995-01-02 | 0900 32 16 8 | 13b | 19 |

# Index

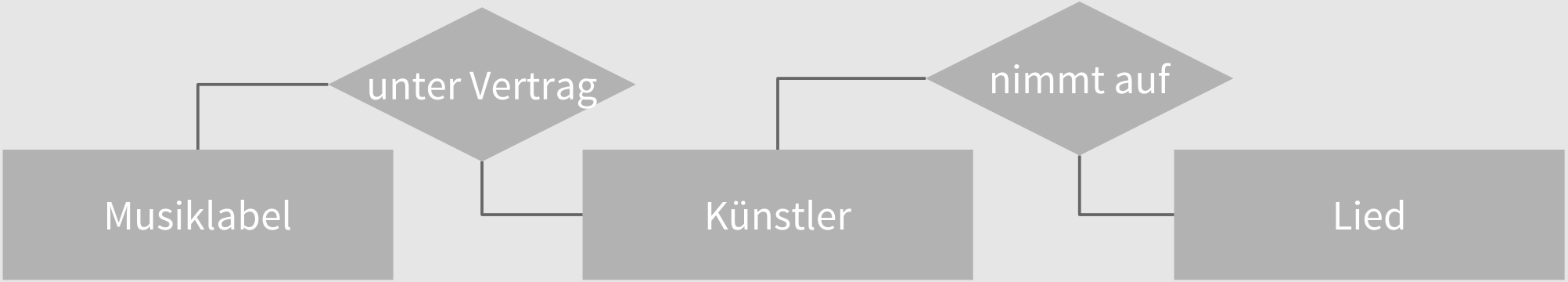

- vgl. Index / Stichwortverzeichnis in Nachschlagewerken
- Nach bestimmten Kriterien im Voraus angelegte Metadaten, um eine spätere Suche zu beschleunigen
- Index-Inhalt wird vom DBMS selbständig verwaltet
- Vorhandensein eines Index ist aber Aufgabe der Entwickler*innen!
- Manche Abfragen können auch rein aus dem Index beantwortet werden


I think we've hit MySQL's limits, it's simply too slow... Let's face it, the amount of data is colossal!
Hum...
So I was thinking we should switch to NoSQL. I've prepared a 3-month battle plan. Here, look!
Framework refactoring, training workshops, internal tool updates...
Wait a minute, let me just check something...


Just as I thought...
Here you go. It's called an index, and it just made your queries 20 times faster...
Dude, missing indexes? Seriously...
CommitStrip.com

# Verknüpfte Tabellen und Joins

- Bereits beim Abruf werden mehrere Tabellen in Beziehung gesetzt
- Beispiel Verknüpfung:
  „Finde alle Musikstücke, die unter einem bestimmten Label produziert wurden“
  (Ich hole nur die Musikstücke, so wie sie in der Tabelle stehen)
- Beispiel Join:
  „Hole alle Lieder und zugehörige Künstler- und Labelnamen“
  (ich hole Sachen am Stück, die gar nicht in einer einzigen Tabelle stehen, und sogar dann, wenn kein Künstler oder kein Label bekannt ist)



# SQL Befehle im Alltag (Überblick)

- Schema oder Datenbankserver verwalten, Teilbereich „DDL“
  - `SHOW SCHEMAS`, `USE`, `SHOW TABLES` zur allgemeinen Orientierung auf einem Server
  - `CREATE SCHEMA` erzeugt Datenbanken
  - `CREATE TABLE` erzeugt Tabellen
- Auf Daten arbeiten (vgl. „CRUD“ in voriger Vorlesung), Teilbereich „DML“
  - `INSERT INTO` fügt Einträge in eine Tabelle ein
  - `SELECT` ruft Einträge aus einer Tabelle ab
  - `UPDATE` ändert Einträge in einer Tabelle
  - `DELETE` löscht Einträge
- (Wie der Titel nahelegt: ALLTAG. Es gibt Haufenweise weitere Kommandos)

# SQL Befehle: Allgemeine Orientierung

- Oft die ersten paar Kommandos auf einem unbekannten Server!
- `SHOW SCHEMAS;`
  → Was gibt es auf diesem Server für Schemas (Datenbanken)
- `USE db_name;`
  → Alle folgenden Kommandos beziehen sich auf db_name
- `SHOW TABLES;`
  → Liste aller Tabellen in einem Schema
- `SHOW COLUMNS FROM table_name;`
  → aufschlüsselung der Struktur von table_name

# CREATE SCHEMA

- `CREATE SCHEMA schema_name;`
  Erzeugt eine leere Datenbank namens schema_name.
- Ja das war's schon.
  Hier ist ein Katzenbild
  für den Rest der Folie:
- Ok, man kann noch encoding angeben. Nehmt UTF-8!

```
CREATE SCHEMA schema_name CHARACTER SET utf-8;
```

# CREATE TABLE

- Erzeugt eine Tabelle (Beispiel):

```
CREATE TABLE `musiclabel` (
   `id` INT NOT NULL AUTO_INCREMENT,
   `name` VARCHAR(255) NOT NULL,
   `established` DATE NULL,
   PRIMARY KEY (`id`)
);
```

**Tabellen-Name**

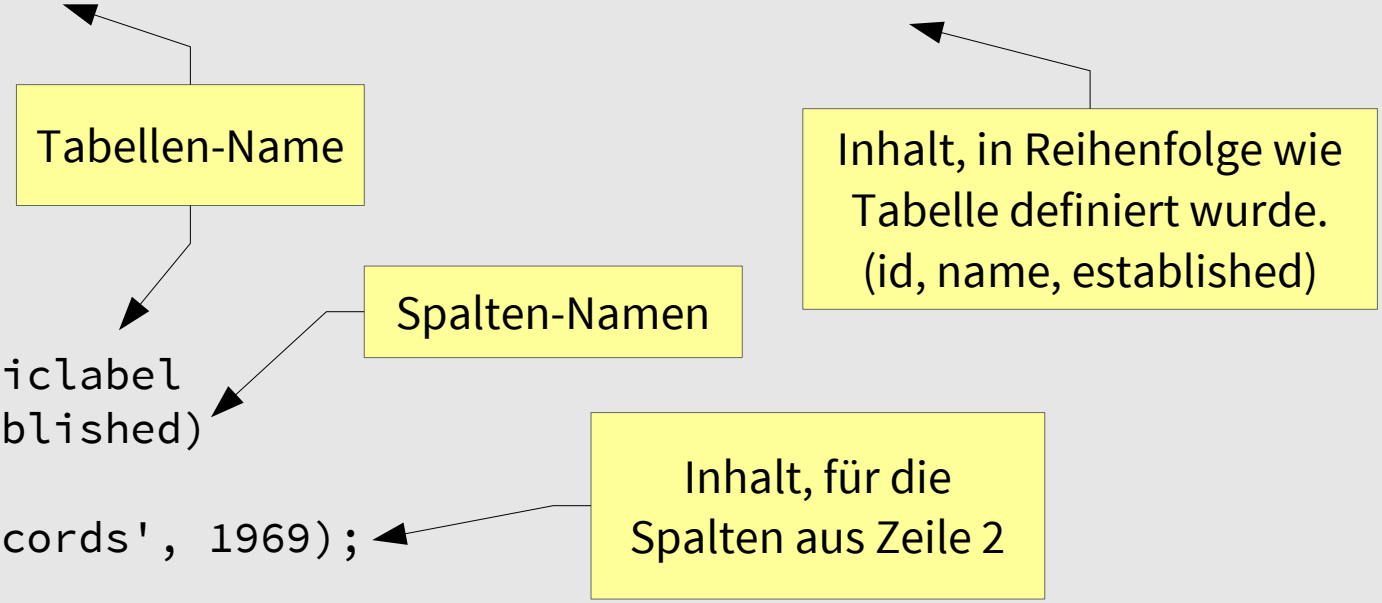
**Spalten:**
Name (z.B. id)
Typ (z.B. INT)
Weitere Eigenschaften
(z.B. AUTO_INCREMENT)

Primärschlüssel ist 'id'

- Details:
  http://dev.mysql.com/doc/refman/5.6/en/create-table.html

# INSERT INTO

- `INSERT INTO musiclabel VALUES (1, 'United Records', 1969);`




```
INSERT INTO musiclabel
    (name, established)
  VALUES
    ('United Records', 1969);
```

- Details: http://dev.mysql.com/doc/refman/5.6/en/insert.html

# SELECT (1 Tabelle)

- Einfacher select:
  ```
  SELECT * FROM tracks;
  ```
- Nur bestimmte Attribute abfragen:
  ```
  SELECT title, duration FROM tracks;
  ```
- Select mit Sortierung
  ```
  SELECT * FROM tracks ORDER BY title;
  ```
- Select mit Bedingung bzw. Filter
  ```
  SELECT * FROM tracks WHERE duration > 240;
  ```
- Select mit weiteren Funktionen (hier: Länge aller Tracks aufummiert, je artist)
  ```
  SELECT artist_id, SUM(duration) FROM tracks GROUP BY
  artist_id;
  ```

* = Alle Attribute

Tabellen-Name

SELECT ist das wichtigste Kommando in SQL

# SELECT (mehrere Tabellen)

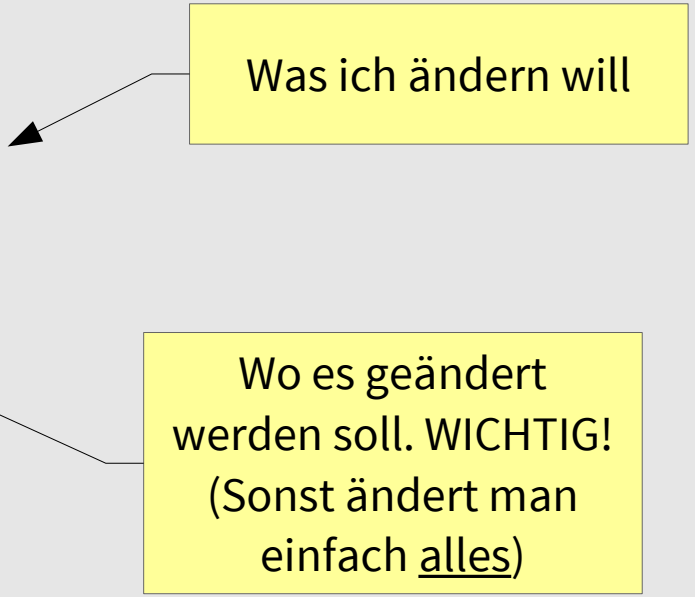
- Falsch: `SELECT artist.name, track.name FROM artists, tracks;`
  → WTF?!
  - „Kartesisches Produkt“ - alle Zeilen werden mit allen anderen Zeilen kombiniert. Ohne Filter unbrauchbar. Achtung: Riesig.
- Richtig:

```
SELECT
  artist.name, track.name
  FROM artists, tracks
  WHERE artists.id = track.artist_id;
```

„Ich mach' ein kartesisches Produkt und filtere da die sinnvollen Kombinationen raus“

# UPDATE

```
UPDATE tracks
    SET duration=182
    WHERE id=124;
```



# DELETE

- Alle Tracks eines Artists löschen

```
DELETE FROM tracks WHERE artist_id = 12;
```

# Musik-Beispiel als SQL Befehle (diesmal zum verstehen)

```sql
CREATE SCHEMA `music` ;
USE `music` ;

CREATE TABLE `musiclabel` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `established` DATE NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `artist` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `musiclabel_id` INT NOT NULL,
  `artist_name` VARCHAR(255) NOT NULL,
  `common_name` VARCHAR(255) NULL,
  `birthday` DATE NULL,
  `hometown` VARCHAR(255) NULL,
  PRIMARY KEY (`id`));
```

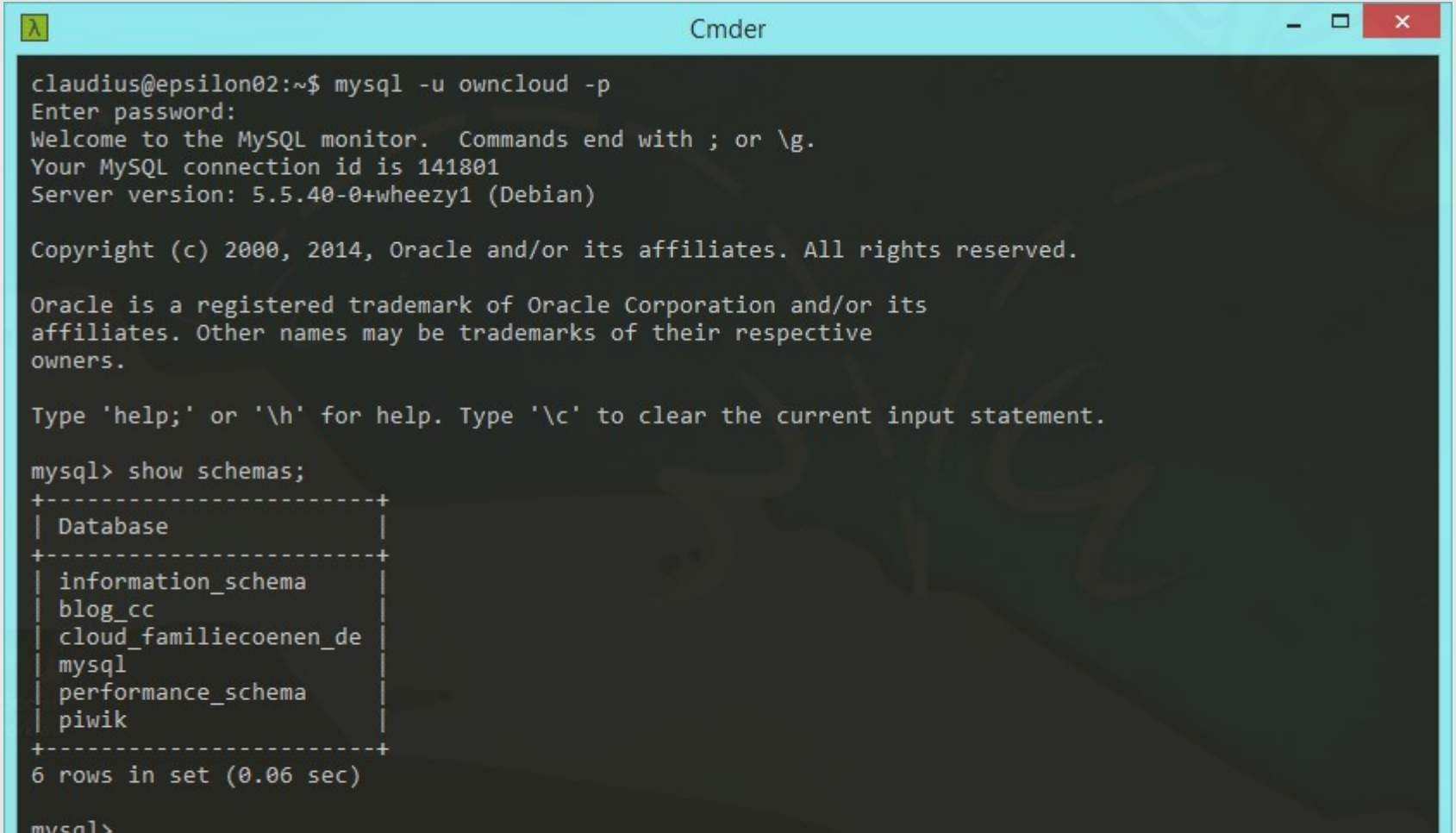
```sql
CREATE TABLE `album` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `artist_id` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  `cover` BLOB NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `track` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `artist_id` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  `duration` DECIMAL(5,2) NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `album_has_track` (
  `album_id` INT NOT NULL,
  `track_id` INT NOT NULL,
  PRIMARY KEY (`album_id`, `track_id`));
```

# Musik-Beispiel als SQL Befehle (copy&pastebar)

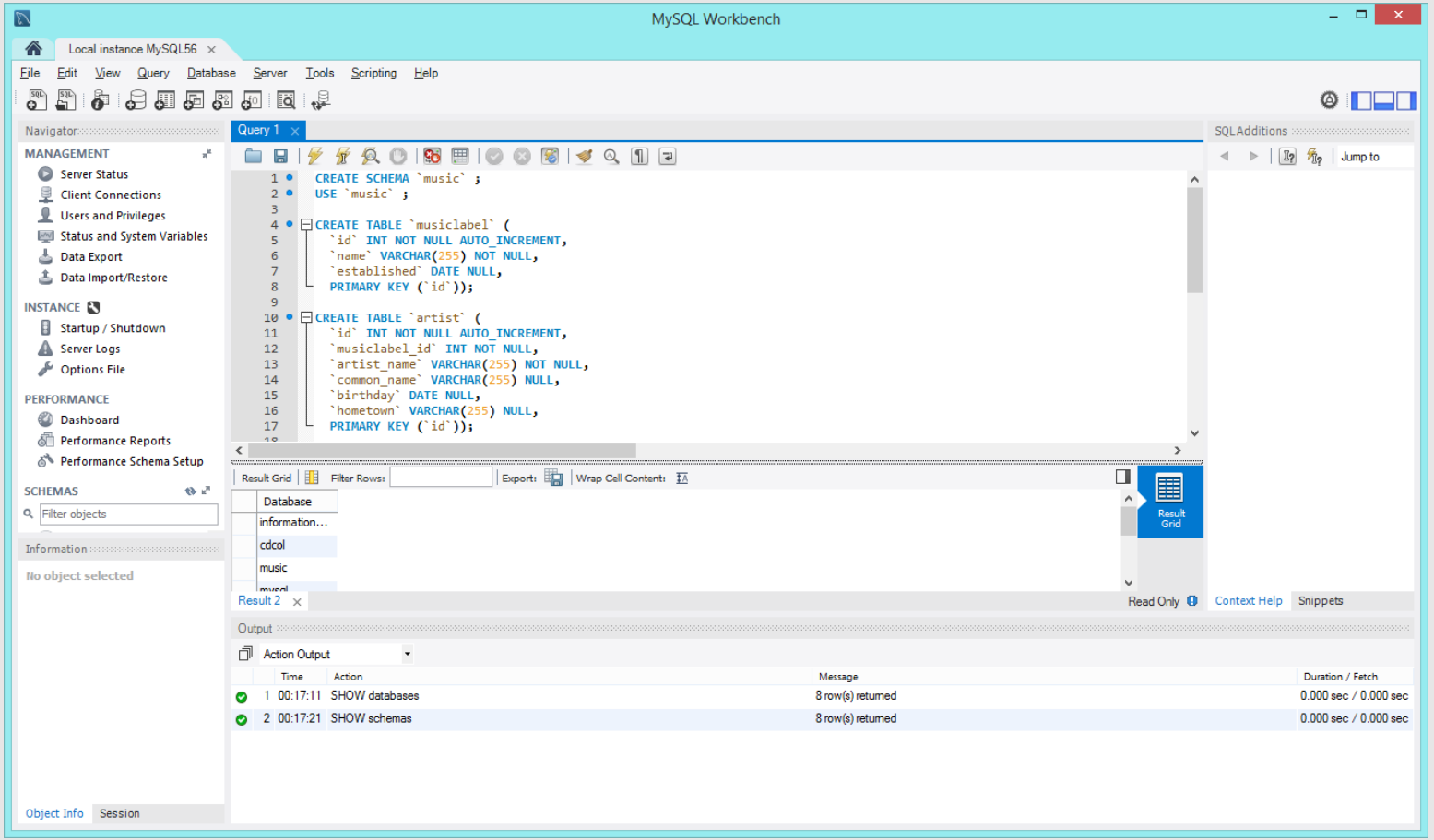
```
CREATE SCHEMA `music` ;
USE `music` ;

CREATE TABLE `musiclabel` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `established` DATE NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `artist` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `musiclabel_id` INT NOT NULL,
  `artist_name` VARCHAR(255) NOT NULL,
  `common_name` VARCHAR(255) NULL,
  `birthday` DATE NULL,
  `hometown` VARCHAR(255) NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `album` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `artist_id` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  `cover` BLOB NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `track` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `artist_id` INT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  `duration` DECIMAL(5,2) NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `album_has_track` (
  `album_id` INT NOT NULL,
  `track_id` INT NOT NULL,
  PRIMARY KEY (`album_id`, `track_id`));
```

# MYSQL TOOLS

# mysql – Command Line Interface



# MySQL Workbench



# ANGRENZENDE THEMEN

# Transaktionen: ACID

- Grundlegend für Datenbankvorgänge ist der Begriff der Transaktion. Für Transaktionen und ihre Verarbeitung gelten folgende Regeln:
  - Sie sind **atomar**, werden also ganz oder gar nicht ausgeführt.
  - Sie stellen in sich **konsistente** Verarbeitungseinheiten dar und lassen eine Datenbasis in einem konsistenten Zustand zurück.
  - Sie laufen quasi **isoliert** ab, d.h. sie bekommen die Wirkung paralleler Transaktionen nicht zu Gesicht.
  - **Dauerhaftigkeit**: Erfolgreiche, durch commit abgeschlossene Transaktionen werden durchgesetzt, d.h. auf dem Plattenspeicher "verewigt''.

Die Anfangsbuchstaben der vier Regeln ergeben
im Englischen das Merkwort: "ACID''

# Transaktionen: Warum?

- Veränderungen, die über mehrere Tabellen hinweg geschehen, kann man nicht in ein UPDATE-Kommando schreiben.
- Transaktionen Lösen das Problem. Man weist die Datenbank an, ein Set von Kommandos am Stück oder gar nicht auszuführen

# Referenzen, Weiterführendes

- Reference Manual:
  http://dev.mysql.com/doc/refman/5.6/en/index.html
- MySQL for Dummies / ISBN 3-8266-3022-X
- PHP&MySQL Novice to Ninja / ISBN 978-0987153081

# AND NOW FOR SOMETHING COMPLETELY DIFFERENT

# Nichts zu verbergen?

- Interessanter Lesestoff: http://wiki.vorratsdatenspeicherung.de/Nichts_zu_verbergen
- In dem Kontext ist auch eine Beschäftigung mit „Privilegien“ sinnvoll Comic-Einstieg: http://www.buzzfeed.com/nathanwpyle/this-teacher-taught-his-class-a-powerful-lesson-about-privil